USDR&E(DDT&E)

SOFTWARE TEST AND EVALUATION PROJECT

SOFTWARE TEST AND EVALUATION MANUAL

VOLUME I

GUIDELINES FOR THE TREATMENT OF SOFTWARE IN TEST AND EVALUATION MASTER PLANS

PREPARED FOR

THE OFFICE OF THE UNDERSECRETARY OF DEFENSE
FOR RESEARCH AND ENGINEERING
DIRECTOR DEFENSE TEST AND EVALUATION

NAVAL AIR DEVELOPMENT CENTER
CONTRACT F33657-82-G-2083

U. S. ARMY MISSILE COMMAND
CONTRACT BOA DAAH01-85-D-A005 D.O. #0008

OCTOBER 1985

THE SOFTWARE TEST AND EVALUATION PROJECT
SCHOOL OF INFORMATION AND COMPUTER SCIENCE
GEORGIA INSTITUTE OF TECHNOLOGY

Report Number: GIT-ICS-85/26
Accession Number: 0558
Title: Software Test and Evaluation Manual,
Volume I: Guidelines for the
Treatment of Software in Test and
Evaluation Master Plans
Contract Number: F33657-82-G-2083 / DAAH01-85-D-A005
Corporate Author or Publisher: Georgia Institute of Technology,
Atlanta, GA 30332
Publication Date: Oct 01, 1985
Pages: 63
Descriptors, Keywords: Software Test Evaluation Defense
System Acquisition Master Plan TEMP

GIT-ICS-85/26

USDR&E (DDT&E)

Software Test and Evaluation Project

Software Test and Evaluation Manual

Volume I

Guidelines for the Treatment of Software

in Test and Evaluation Master Plans

Prepared for

The Office of the Undersecretary of Defense
for Research and Engineering (USDR&E)

Director Defense Test and Evaluation (DDT&E)

October 1985

Prepared by the Software Test and Evaluation Project, Georgia Institute of Technology, School of Information and Computer Science, Atlanta, GA 30332 under Naval Air Development Center Contract Number F33657-82-G-2083 and U. S. Army Missile Command Contract Number BOA DAAH01-85-D-A005 D.O. #0008.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GIT-ICS-85/26	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Software Test and Evaluation Manual Volume I Guidelines for the Treatment of Software in Test and Evaluation Master Plans		5. TYPE OF REPORT & PERIOD COVERED Final Report
		6. PERFORMING ORG. REPORT NUMBER GIT-ICS-85/26
7. AUTHOR*(s)* Software Test and Evaluation Project		8. CONTRACT OR GRANT NUMBER*(s)* F33657-82-G-2083 and BOA DAAH01-85-D-A005 D. O. #0008
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Information and Computer Science Georgia Institute of Technology Atlanta, GA 30332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS NADC Warminster, PA 18974	U.S. Army Missile Command Redstone Arsenal, AL 35898-5280	12. REPORT DATE October 1985
		13. NUMBER OF PAGES 63 + iv
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Distribution Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Software Test and Evaluation Manual, defense system acquisition, Software Test and Evaluation Project (STEP), Director Defense Test and Evaluation (OSD/USDR&E), Test and Evaluation Master Plans (TEMPs), mission critical software, software intensive,risk, checklist, program offices, independent test organizations, contractors

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Software Test and Evaluation Manual provides guidance to DoD components in the area of software test and evaluation for defense system acquisition. It was prepared by the Software Test and Evaluation Project (STEP) in response to tasking by the Director Defense Test and Evaluation (OSD/USDR&E).

Volume I is intended to support the review of Test and Evaluation Master Plans (TEMPs) for systems that contain mission critical software components, are

Unclassified (over)

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)* software intensive, or present software testing issues that significantly affect risk. It consists of a checklist or series of questions that are keyed to the major paragraphs of a TEMP, and an accompanying set of explanatory notes that are brief commentaries on the questions and the significance of the possible responses to them.

The primary audience for this manual consists of those individuals who are responsible for evaluating TEMPs. However, it should also prove useful to program offices, independent test organizations, contractors, and others who review, prepare, or provide data for inclusion in TEMPs.

OFFICE OF THE UNDER SECRETARY OF DEFENSE

WASHINGTON, D.C. 20301

RESEARCH AND
ENGINEERING
(DDTE)

18 September 1985

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Software Test and Evaluation Manual

The enclosed Software Test and Evaluation Manual is a two volume reference set prepared by the Software Test and Evaluation Project (STEP) sponsored by this office and is being sent to you for review, comment, and use pending its publication as an official Department of Defense Manual . The goal of the manual is to provide consistent guidelines for the preparation and evaluation of Test and Evaluation Master Plans (TEMPS) for major software intensive systems containing mission critical computer resources. In addition, this manual summarizes the technological, organizational, and contractual issues that should be addressed during the development and execution of a comprehensive and effective program of software test and evaluation.

STEP was initiated by this office in 1981 to investigate the feasibility of and make recommendations for improving Department of Defense Policy and Guidance for Software Test and Evaluation. The principal policy vehicle for test and evaluation is DoD Directive 5000.3, which also specifies the format and function of the TEMP. The present manual is a key tool in an on-going program aimed at improving the test and evaluation of major systems through improved acquisition management and risk reduction procedures. This manual contains important implementing tools which will be applied in support of existing and planned policy for the test and evaluation of Software-Intensive Systems.

Subsequent reissues of this manual will reflect changes to DoD Directive 5000.3, implementing service regulations and standards, as well as improvements made possible by an ever-advancing State-of-the-Art in Software Test and Evaluation.

Comments and inquires are requested and may be addressed to:

OUSDRE,DDTE
Room 3E1060, The Pentagon
Washington, D. C. 20301-3110

C K Watt
CHARLES K. WATT
Acting Director
Defense Test and Evaluation

Enclosure

Contents

Section . . . . . Page

Section 1. Notes to the User 1

Section 2. TEMP Outline, Software Checklist and Explanatory Notes . 3

Part I. Description 4

1. Mission 4
   Note 0 5

2. System 6
   Note 1 7
   Note 2 9
   Note 3 11

   a. Key Functions 12
   Note 4 13

   b. Interfaces 14
   Note 5 15

   c. Unique Characteristics 16
   Note 6 17

3. Required Operational Characteristics 20

4. Required Technical Characteristics 20
   Note 7 21

5. Critical T&E Issues 26
   a. Technical Issues 26
   b. Operational Issues 26
   Note 8 27

Part II. Program Summary 28

1. Management 28
   Note 9 29
   Note 10 29
   Note 11 31
   Note 12 31

2. Integrated Schedule 32
   Note 13 33
   Note 14 33

Section Page

Part III. DT&E Outline 34

1. DT&E to Date 34
Note 15 35
Note 16 37
Note 17 39

2. Future DT&E 40
a. Equipment Description 40
b. DT&E Objectives 40
c. DT&E Events/Scope of Testing/Basic Scenarios 40

3. Critical DT&E Items 40
Note 18 41

Part IV. OT&E Outline 42

1. OT&E to Date 42
Note 19 43

2. Future OT&E 44
a. Equipment Description 44
b. OT&E Objectives 44
c. OT&E Events/Scope of Testing/Basic Scenarios 44

3. Critical OT&E Items 44
Note 20 45

Part V. PAT&E 46
Note 21 47

Part VI. Special Resource Summary 48

1. Test Articles 48

2. Special Support Requirements 48
Note 22 49
Note 23 49
Note 24 49

Section 3. References and Glossary 51

References 51
Glossary 53

Index 59

Software Test and Evaluation Manual

Volume I

Guidelines for the Treatment of Software in Test and Evaluation Master Plans

Section 1
Notes to the User

This manual is intended to support the review of Test and Evaluation Master Plans (TEMPs) for systems that:

1. contain mission critical software components,

2. are software intensive, or

3. present software testing issues that significantly affect risk.

The primary audience for this manual consists of those individuals who are responsible for evaluating TEMPs. However, this manual should also prove useful to program offices, independent test organizations, contractors, and others who review, prepare, or provide data for inclusion in TEMPs.

The overall organization of Section 2 of this volume is keyed to the TEMP structure defined by DoD Directive 5000.3, "Test and Evaluation" (December, 1979; Enclosure 2). It should be possible to evaluate a TEMP using this portion of the document as a roadmap of the software issues that may arise. Section 2 contains:

> The TEMP and Software Checklist: The Checklist is a series of questions keyed to the major paragraphs and sections of a TEMP. In format, the questions are phrased to permit simple "yes" or "no" answers. In practice, it is not necessary that every checklist question be answered by the TEMP. That is, a checklist question may be inapplicable to the system being reviewed. In other cases, checklist questions that are not answered by the TEMP may indicate deficiencies.
>
> Explanatory Notes: The Notes are brief commentaries on the checklist questions and the significance of the possible responses to them. The manual user needs to be aware that the Notes are of a general nature and may not accurately reflect the intricacies of the technology used in a given system. Nevertheless, the Notes describe issues that a software engineer would raise in evaluating whether or not the software has demonstrated, through extensive usage, testing, and repair, that it meets design and user requirements. Therefore, these issues should be of primary concern when evaluating a TEMP.

Each of the checklist questions references an explanatory note. The notes appear on the page opposite the question that references them. This layout has been chosen with the goal of minimizing the amount of "paging" required when using this manual.

Section 3 consists of the References and Glossary. The Checklist and Notes are not intended to be a "textbook" in software testing. Variations in terminology and basic software definitions are accounted for in the Glossary. Detailed discussions of technical matters that are mentioned in the Notes are contained in the References.

Section 2

TEMP Outline, Software Checklist

and

Explanatory Notes

TEMP Section/Subsection		Checklist Questions
Part I - Description		
1. Mission. This section summarizes operational need, mission to be accomplished and planned operational environment and relates directly to the Mission Element Need Statement (MENS) and planned system operational concept.	___	**(Note 0)**

Note 0:

The description of the mission to be performed should be a statement of need and operational concept. Software specific references are generally not appropriate in this section. A possible exception may occur if the only way to express the system operational concept is in software-oriented terminology.

References for Note 0: [Redwine, Chapter II]

<u>TEMP Section/Subsection</u>		<u>Checklist Questions</u>
<u>Part I - Description</u>		
2. <u>System</u>. This section is a brief description of the system including discussions of key functions, interfaces, and unique characteristics.	___	Does the system contain mission critical computer resources? **(Note 1)**

Note 1:

The term "Mission Critical Computer Resources (MCCR)" is defined by Section 908 of the FY 1982 Defense Authorization Act. MCCR include automatic data processing equipment or services whose functions are critical to support:

— intelligence activities
— cryptologic activities related to national security
— command and control of military forces
— equipment that is an integral part of a weapon or weapon system
— the direct fulfillment of military or intelligence missions.

Computers for MCCR applications are exempted from the provisions of the Brooks Act (P.L. 89-306). Acquisition policy for MCCR is defined by DoD Directive 5000.29 ("Management of Computer Resources in Major Defense Systems").

From a technical point of view, knowing whether or not the application contains MCCR is often critical in determining the role that software plays in controlling systems functions. Therefore, it is essential that the TEMP be sufficiently detailed to answer this question.

If the system contains an MCCR application, the TEMP should allow an easy determination of how critical software is to the essential mission functions. One possibility is that mission objectives are achieved by special-purpose computers or circuitry in which software instructions play no important role. These situations are rare, since a key role of MCCR technology is to provide for systems that are easily changeable (e.g., to adapt to changing threats). Another possibility is that the system contains an MCCR application and that software plays a significant role.

The kind of MCCR application may point to areas in which special T&E considerations should be taken into account. For example, there may be secure functions whose implementations should be certified by the DoD Security Center.

The nature of an MCCR application may also lead to management issues that affect risk. For example, the use of MCCRs in some applications leads to specialized Service standards and regulations; these may by themselves give rise to T&E issues. The same considerations may also affect inter-agency agreements for scheduling and budgeting and down-stream problems such as test data-sharing and test-bed availability.

Even if the MCCR application involves neither software intensive applications (cf. Note 2) nor mission critical software (cf. Note 3), there may be significant software T&E issues. For example, in many real-time systems, a degree of fault-tolerance can be gained by software (e.g., providing for graceful degradation of communicating processors). System designers may have ignored such opportunities and, as a result, the system may have few software features that protect users from faulty hardware. In such an instance, the lack of software features should be treated as a design flaw that will ultimately reduce system availability.

References for Note 1: [Grove], [Redwine, Chapter II], [STEP1], [STEP5, Nelson], [STEP5, Stewart]

TEMP Section/Subsection		Checklist Questions
Part I - Description		
2. System. This section is a brief description of the system including discussions of key functions, interfaces, and unique characteristics.	___	Is the system software intensive? **(Note 2)**

Note 2:

If the system does not contain MCCR, the software may still contribute to overall risk. (Non-MCCR applications are generally not subject to the policies governing MCCR systems; see Note 1.) This is often the case when the bulk of system development is devoted to software. An example of such a system might be an automatic data processing (ADP) application in which the system hardware is an off-the-shelf commercial product, but the software needs to be developed to meet military requirements. In these situations, the operational characteristics of the commercial hardware cannot be transferred to the software (e.g., ease of maintenance for the hardware does not alleviate potential maintenance problems for the software).

Other sources of risk arise when the software is likely to be difficult to design or requires the use of new or undemonstrated software engineering practices and tools.

Even if the system does not contain mission critical computer resources, it is software intensive if it contains software that:

— dominates the development budget

— is large or complex relative to overall system size and complexity.

It is usually prudent to treat software intensive systems as if they contained mission critical computers and software. For the T&E community, these systems pose special problems. The framers of the system development plans and the TEMP are responsible for ensuring that proper consideration is given to the testing of the software in these systems.

One aspect of the TEMP that needs special attention for software intensive systems — especially for non-MCCR systems — is the description of functional capabilities that will be demonstrated by testing. For non-MCCR systems, these may be related only remotely to operational mission objectives. Therefore, these capabilities are easy to overlook at the system level.

A careful review of required system characteristics and critical interfaces (see Notes 5 and 7 below) to determine those that are traceable to software requirements may be helpful. Functional areas in which non-MCCR software plays an important role include the following:

— data base management
— communication and networking
— CAD/CAM and support software development
— training and simulation
— computer graphics and human interfaces
— decision support.

A system having functional capabilities in any of these areas — or having significant interfaces with systems that provide these capabilities — is probably software intensive. The test and evaluation of the software in these systems should be planned and managed as if the system contained mission critical software.

References for Note 2: [Bunyard], [STEP1]

TEMP Section/Subsection		Checklist Questions
Part I - Description		
2. System. This section is a brief description of the system including discussions of key functions, interfaces, and unique characteristics.	___	Does the software implement critical functions? **(Note 3)**

Note 3:

An adequate risk assessment requires that this question be answered at the system level. If the system contains mission critical computer resources (see Note 1) or if it is software intensive (see Note 2), then the capabilities to be performed by the software should be explicitly cited. Failure to highlight software-implemented mission critical functions may lead to tests that do not adequately assess conformance to technical and operational objectives.

In addition, the lack of software-implemented functions may in some circumstances be questioned. For example, a system that is supposed to be capable of rapid adaptation but has only hardware implementation of its critical functions may exhibit unacceptable availability characteristics in operation.

The paramount issue that arises from a "YES" answer to this question is whether the software has been given balanced treatment with other critical system components. An early determination of the software-implemented functions allows for careful design of testable software requirements and specifications and the development of a systematic approach to software testing. Experience has shown that when these considerations are pushed later into the development/acquisition process, latent problems with the software are more difficult to eliminate and the resulting systems are less well-suited to their objectives.

To emphasize the balance that should be sought between the software and the hardware components that implement critical functions, the term critical software component will be used throughout this manual. A critical software component is any portion of a computer program, any computer program, or any collection of computer programs that fulfills a requirement for a Key Function as described in the TEMP.

References for Note 3: [Redwine, Chapter III], [STEP1], [STEP3, Part 2]

TEMP Section/Subsection | Checklist Questions

Part I - Description

2. System.

a. Key Functions: these are functions that enable the system to accomplish its operational mission; this description may include a mission/function matrix relating primary functional capabilities that must be demonstrated by testing to the mission to be performed and the concept of operation.

___ Does the Mission/Function matrix identify primary functional capabilities to be implemented by the software? **(Note 4)**

___ Are the identified functions implemented in software:

- ___ New? **(Note 4a)**
- ___ Automation or modification of existing capabilities? **(Note 4b)**
- ___ Mature? **(Note 4c)**

<u>Note 4</u>:

The mission/function matrix (or equivalent narrative) is the primary source of information about how the capabilities have been partitioned between hardware and software. These partitions will be important in determining required characteristics, in defining error/failure categories, and in isolating and correcting deficiencies noted during testing. Therefore, it may be important to determine that proper engineering studies have led to the establishment of these partitions.

An understanding of the sources of risk in each of the software-implemented functions identified in the mission/function matrix is an essential part of the overall risk assessment. Some typical risk drivers are those which influence the maturity of the software.

a. <u>New Function</u>: New software functions generally represent the highest risk, since they involve not only the design of software, but also the use of new concepts, theories and algorithms. These functions are often found in applications of emerging critical technologies such as artificial intelligence, distributed data processing, or ultra-reliable computing. Often, these functions have only been demonstrated in the laboratory and no operator personnel have been exposed to the functions under realistic conditions. Questions of suitability and performance are typical for these functions and the early involvement of users and operational testers is encouraged. Risk reduction procedures such as prototyping, simulation, and evolutionary acquisition may also be appropriate.

b. <u>Automation or Modification of Existing Capabilities</u>: The transition from manual functions to automated functions is notoriously hard to manage. Functions performed by humans are usually difficult to formally describe; it is therefore hard to test the conformance of the automated capability to a set of technical specifications. On the other hand, the functions themselves are generally mature, so suitability in an operational setting is not a critical issue. There should be a clear plan for determining the extent to which the previously manual capability has been faithfully reproduced.

c. <u>Mature Software</u>: It is also possible that the software implementation of the function is mature. This is generally the lowest risk implementation of the function. Many modern software design methodologies promote the reuse of software as a way of improving overall software quality. If reusable software is included, then the TEMP should discuss the extent to which previous testing can be applied to the current implementation.

<u>References for Note 4</u>: [Redwine, Chapter II]

TEMP Section/Subsection		Checklist Questions
Part I - Description		
2. System.		
b. Interfaces: these are points of interaction with other systems that are required to accomplish the mission.	___	Is software important to the interfaces? **(Note 5)**
	___	Do the interfaces have software implications? **(Note 5)**

Note 5:

Electronic interfaces between systems are frequently responsible for interoperability and communication. The impact of these interfaces on the software should be discussed. It is unlikely that electronic interfaces can be designed without significant software involvement.

For example, the software may be important to the successful implementation of the interface. Even if the interface hardware is off-the-shelf, the software is likely to be unique to the current system and is therefore of higher risk. If the interface corresponds to a key function, then the software should be treated as mission-critical (cf. Note 3).

Further, the interfaces may place additional requirements on the software. For example, communicating systems of processors frequently require specialized formatting of data for transmission. Such data formatting functions are usually software-implemented. In many applications, there are industry standards that can be invoked. If the interfaces place requirements on the software, these should be discussed and (in an appropriate section) a plan should be presented for ensuring that the requirements have been met.

References for Note 5: [Chou]

TEMP Section/Subsection	Checklist Questions
Part I - Description	
2. System.	
c. Unique Characteristics: these are aspects of the system that make it better than or different from alternative systems, or that lead to special test requirements.	___ Does the system use software engineering technology that:
	___ Affects risk? **(Note 6a)**
	___ Has lifecycle impact? **(Note 6b)**
	___ Are there any software characteristics or aspects of the software application that distinguish the system from alternative systems? **(Note 6c)**

Note 6:

The tools, methodologies and engineering techniques that are used to produce operational software are key sources of risk in a development program. Software contractors, for example, frequently compete on the basis of software "methodologies" that combine management approaches and technologies in unique ways. Therefore, some novelty in the system is to be expected. However, the TEMP should discuss those novel aspects that might affect an overall evaluation.

a. Affect Risk: A principal source of risk is the toolset or support software used to construct the operational software. Examples of support software and software tools range from any of the relatively mature text editors that can be purchased off-the-shelf to the special purpose compilers that produce object code for the operational or target computers. The choice of high order language (HOL) used in the project is a frequent source of risk that may contaminate the whole system. In particular, new HOL's increase risk along several dimensions:

   1. a "learning curve" effect limits the productivity of design and implementation teams during early project phases,

   2. the immaturity of the HOL compiler increases the likelihood that software errors may be introduced during the implementation or that inefficiencies in compiler-generated code may limit the ability of the system to meet performance goals,

   3. the suitability of the new HOL for the application may be undemonstrated.

   Sometimes independent risk reduction is available for support software. For example, new Jovial and Ada compilers should have been DoD certified before their use on the project. Evaluation of support software should also specify what has not been assessed; for instance, performance parameters of compilers are not typically addressed during compiler validation. When these concerns are important to the overall mission of the system, appropriate T&E should be planned.

b. Lifecycle Impact: A common source of operational software problems is the difficulty of maintaining and supporting the software once it is deployed. The technology used to design and implement the software may significantly affect this ability. Danger signals may include the use of proprietary tools and techniques that will not be available to engineers after system delivery. Alternatively, there may be unique aspects of the design effort that positively affect subsequent lifecycle cost and effort. One approach to reducing long-term lifecycle risks is to enforce the use of common technology throughout the development and operation of the software. It is not uncommon for the project office to supply tools and support software GFE to the contractor to ensure commonality. However, care should be exercised to avoid Government liability in cases of inadequate Government furnished tools.

TEMP Section/Subsection	Checklist Questions
Part I - Description	
2. System.	
c. Unique Characteristics: these are aspects of the system that make it better than or different from alternative systems, or that lead to special test requirements.	___ Does the system use software engineering technology that: ___ Affects risk? **(Note 6a)** ___ Has lifecycle impact? **(Note 6b)** ___ Are there any software characteristics or aspects of the software application that distinguish the system from alternative systems? **(Note 6c)**

Note 6 (cont.):

Ideally, lifecycle characteristics of operational significance should be listed as required characteristics of the system (cf. Note 7) and tests should be planned to address the issues that arise from these characteristics.

c. Distinguish from Other Systems: Two types of distinguishing characteristics are important: those that differentiate the given application from all others and those that distinguish the current generation of the system from its predecessors.

Certain applications (e.g., those with nuclear implications) are subject to requirements and certifications that are not levied on other applications. Care should be taken to determine the extent to which software is represented in these applications. The approach to software testing taken in some of these applications may be questioned. The descriptions of independent evaluations, validations and certifications for these applications should define terms. Typical questions to be raised are the following:

Does the cost of testing balance the "cost" of failure in this application?

Does the testing approach require new or undemonstrated software or hardware technology that in itself raises risk? (cf. Note 6a)

The system may also be unique to the extent that the software is responsible for extreme performance or reliability goals. A proposed system may raise quantitative requirements by one or more orders of magnitude. During early program phases, clear demonstration that the goals can be met should be provided. Adequate demonstrations can be obtained by proof-of-concept prototyping, analytical studies, and in some instances by non-standard acquisition strategies (e.g., evolutionary or P3I).

References for Note 6: [Redwine, Chapter III], [STEP3, Part 2]

TEMP Section/Subsection	Checklist Questions
Part I - Description	
3. Required Operational Characteristics. Key operational effectiveness and suitability characteristics, goals and thresholds.	
4. Required Technical Characteristics. Key technical characteristics, performance goals, and thresholds.	
Note: The characteristics listed in 3-4 above should include, but not be limited to, the characteristics identified in the Decision Milestone documentation. Clearly define these characteristics, particularly in the areas of reliability, availability, and maintainability. Indicate program milestones at which the thresholds will be or have been demonstrated.	___ Are there operational or technical characteristics that are: ___ Unique to software? **(Note 7)** ___ May have been overlooked? **(Note 7)**

Note 7:

A necessary component of system level test planning is the definition of goals and thresholds for the critical software components. A TEMP for a system that contains mission critical software should also describe the primary indicators of the software's:

— conformance to written specifications (required technical characteristics)

— operational suitability and effectiveness (required operational characteristics).

A TEMP that fails to define these characteristics for critical software components is deficient in that it has failed to set goals and thresholds for the characteristics of mission critical functions.

Special care should be taken to ensure that required software characteristics have been presented. A major reason for omitting references to software in the required characteristics is that the software characteristics have aspects that are unique to software technology. The framer of the TEMP may have little experience in judging the relative importance of these characteristics.

Another reason for not including software characteristics in the TEMP is that they do not fit cleanly into the technical/operational definitions. In fact, one distinguishing feature of software is that the goals and thresholds of interest may blur the distinction between technical and operational characteristics. In developing test plans and schedules, care must be exercised to ensure that software characteristics are evaluated at the appropriate stage of system development rather than at arbitrarily imposed milestones. It is a mistake to wait until the hardware and software are integrated to resolve outstanding software test issues (cf. Note 8). For instance, some operational parameters associated with the software can be reliably determined during development testing but cannot be measured directly during operational testing.

Early evaluation of software characteristics should be an integral part of the development process. Late treatment of the software opens the following problems:

— error masking: hardware and software errors may in some instances mask each other, complicating RAM analysis.

— error partitioning: without a reliable estimate of software failure rates, the partitioning of operational errors/failures into hardware, operator, and software failures may be subjective and inexact.

TEMP Section/Subsection		Checklist Questions
Part I - Description		
3. Required Operational Characteristics. Key operational effectiveness and suitability characteristics, goals and thresholds.		
4. Required Technical Characteristics. Key technical characteristics, performance goals, and thresholds.		
Note: The characteristics listed in 3-4 above should include, but not be limited to, the characteristics identified in the Decision Milestone documentation. Clearly define these characteristics, particularly in the areas of reliability, availability, and maintainability. Indicate program milestones at which the thresholds will be or have been demonstrated.	___	Are there operational or technical characteristics that are: ___ Unique to software? **(Note 7)** ___ May have been overlooked? **(Note 7)**

Note 7 (cont.):

As a guide to locating and evaluating required software characteristics, the following examples should be taken into account.

1. Reliability: This characteristic is often a key indicator of software suitability. It is very important to choose metrics and measurement criteria that adequately reflect software reliability. On the other hand, it should be recognized that software reliability has unique aspects. Classical time-dependent reliability theory may not apply to software. Probability distributions are notoriously ineffective in describing failure rates for software except in cases where the true operational distributions of the inputs are known. Observing software failures in integrated hardware/software systems is difficult. Low reliability estimates for software that implements critical functions should be questioned. If the software is duplicated in multiple platforms, then failure rates are multiplicative (since many instruction executions take place), and even very low failure rates can result in significantly many operational mission failures. The use of tests that exercise and stress software components and demonstrate functional behavior in simulated operational environments should be considered.

2. Availability and Maintainability: Hardware-oriented definitions of availability and maintainability are seldom satisfactory for software. Parameters such as mean logistic down time that take into account spare parts requirements and transportation time do not adequately capture software availability. Maintainability of software incorporates repair and re-engineering. Usually maintenance is carried out at a Post Deployment Software Support (PDSS) facility and factors limiting mean time to repair tend to revolve around communications and the labor-intensiveness of the maintenance process.

3. Human Factors: As an indicator of operational suitability, this aspect can be evaluated early. The use of simulators, prototype hardware, and operator personnel can give reliable indications of software suitability in the laboratory setting. Early determination of deficiencies allows correction through redesign of the software. Late detection of unsuitable human factors in the software can raise the cost of correction by one or more orders of magnitude.

4. Performance: The operational performance parameters may be determined very early in the development process by technical software characteristics. For example, the ability of a system to track and engage multiple targets may be limited by the precision and accuracy of the algorithms used in software design, the efficiency of a frequently executed mathematical subroutine, or by the size of a software buffer. Another performance threshold may be the ability of the system to operate in a degraded mode above a certain threshold. Such a performance parameter may be solely due to the "robustness" of the software, a technical characteristic that indicates how well the software operates when its input does not satisfy the input specifications.

TEMP Section/Subsection	Checklist Questions
Part I - Description	
3. Required Operational Characteristics. Key operational effectiveness and suitability characteristics, goals and thresholds.	
4. Required Technical Characteristics. Key technical characteristics, performance goals, and thresholds.	
Note: The characteristics listed in 3-4 above should include, but not be limited to, the characteristics identified in the Decision Milestone documentation. Clearly define these characteristics, particularly in the areas of reliability, availability, and maintainability. Indicate program milestones at which the thresholds will be or have been demonstrated.	___ Are there operational or technical characteristics that are: ___ Unique to software? **(Note 7)** ___ May have been overlooked? **(Note 7)**

Note 7 (cont.):

In summary, "performance" — as the term is commonly applied to software systems — includes the quantified efficiency or capacity of the programs. Even though the primary (system-level) characteristic is operational, the best predictor of software performance is usually technical.

References for Note 7: [AFOTECIII], [IBM], [Meyers], [Perlis]

TEMP Section/Subsection	Checklist Questions
Part I - Description	
5. Critical T&E Issues a. Technical Issues: key areas of technological or engineering risk that must be addressed by testing. b. Operational Issues: key operational effectiveness or suitability issues that must be addressed by testing.	___ Do the required software characteristics raise unique or easily overlooked T&E issues? **(Note 8)**

Note 8:

A critical software T&E issue is any aspect of the software system's capability that must be questioned before a system's overall worth can be estimated. The software issues are of primary importance in reaching a decision as to whether the system should advance into later programmatic phases. This decision is to be based in part on the determination that the goals and thresholds defined for the required software characteristics have been met and, in any case, should be based on an assessment that software and hardware risk have been balanced by the past and future T&E.

In addition to other discussions of software issues that may be present in the TEMP, goals and thresholds should be associated with each issue that will be addressed by testing. This will be the basis for judging the effectiveness of more detailed software test plans and will provide the framework for interpretation of software test results.

For example, a critical software T&E issue may be that the maintainability of the software is to be validated (cf. Note 7). Maintenance of operational software may require:

1. a PDSS or similar support facility that is adequate for the re-engineering effort that may be required during maintenance

2. a logistics support network for transferring maintained software from the PDSS to the fielded system

3. the skilled human resources necessary to re-engineer a large software system under severe scheduling constraints.

The validation of software maintainability as an operational parameter (e.g., is the MTTR for critical software components sufficiently small to ensure that system availability goals can be met?) leads to the following question: is the test environment representative of the environment in which the software will actually be maintained? The T&E outlines should provide a detailed answer to this question. In the case of validating software maintainability, the PDSS personnel should conduct the tests. Use of PDSS personnel meets one objective of an operational test — use of typical operator personnel in a typical operational environment — and also ensures a realistic estimation of the maintainability characteristic.

As pointed out in Note 7, quantifiable progress toward goals is the most desirable way of posing a critical issue. However, objective evaluations of progress are oftentimes more important than ad hoc quantification. For example, meeting a time dependent reliability goal $R(t) = 0.97$ for t hours of operation is seldom meaningful for software, and any attempts to provide such a statistical measure should be questioned. On the other hand, achieving an observed mean time between operational mission failure (MTBOMF) of t hours is meaningful and allows analysts to concentrate on validating the realism of the test scenario, the software contribution to observed operational failures, and other issues that help determine the indices of progress for the software.

As further discussed in Note 7, issues should also be defined to expose software uniqueness in the issues and should associate technical or operational meanings to the issues, regardless of the standard (hardware) interpretation.

References for Note 8: [Brown], [STEP1]

TEMP Section/Subsection	Checklist Questions
Part II - Program Summary	
1. Management. Outline the program and T&E management responsibilities of participating organizations. Highlight arrangements between participants for test data sharing, responsibilities for test management decisions, and management interfaces for multiservice T&E efforts. Discuss the adequacy of the planned test periods and schedule to provide confidence in test results.	___ Is there a manager with principal responsibility for software in the project office? In test organizations? **(Note 9)**
	___ Are the project offices and test organizations aware of T&E that will be carried out by parallel organizations? **(Note 10)**
	___ Are the results of tests of software components available to subsequent test groups? **(Note 10)**

Note 9:

Experience has shown that all aspects of software development and testing progress more efficiently when a knowledgable software manager is present and active. Software issues tend to cut across system issues. The program management structure should ensure that software concerns are not left unattended.

References for Note 9: [Brown], [STEP3, Part 2]

Note 10:

A number of DoD and non-DoD organizations carry out testing and validation for both operational and support software (cf. Note 6a). These organizations and the evaluations they carry out include the following:

- DoD Security Center (software security certification)
- National Bureau of Standards (software cryptologic certification for Data Encryption Standard)
- Federal Software Testing Center (testing of compilers and support software for conformance to specifications)
- Ada Joint Program Office, OUSDRE (validation and certification of Ada compilers)
- Air Force Language Control Facility (validation and certification of Jovial compilers, cataloging of tools for Jovial programming environments)
- Product Engineering Services Office, OUSDRE (evaluation of FOT&E for systems in production)

In addition several groups of testers may proceed independently. Contractors may produce test results that are useful to independent Government testers. Development testers may generate results of simulations that provide indications of operational effectiveness to operational testers. Operational test scenarios may be analyzed by development testers to determine software test coverage. In each applicable instance, possibilities for test data sharing and incorporation of independent certifications into the TEMP should be questioned.

References for Note 10: [STEP3, Part 2]

TEMP Section/Subsection		Checklist Questions
Part II - Program Summary		
1. Management. Outline the program and T&E management responsibilities of participating organizations. Highlight arrangements between participants for test data sharing, responsibilities for test management decisions, and management interfaces for multiservice T&E efforts. Discuss the adequacy of the planned test periods and schedule to provide confidence in test results.	___	Is there evidence that available sources of expertise have been explored and that coordination has been carried out with programs designed to assist MCCR software projects? **(Note 11)**
	___	Are management aids (e.g., tools, checklists, guides, and decision support systems being used? **(Note 12)**
	___	
	___	Is there a plan to use electronic mail/communications among participating organizations? **(Note 12)**

Note 11:

The effective utilization of existing software technology, practices, and management techniques may be aided by the assistance of special programs in DoD and the military Services.

Overall responsibility for coordinating software initiatives and technology transition programs in DoD rests with the Director, Computer Software and Systems (OUSDRE). These programs may offer human and technical resources to assist MCCR software development efforts.

Army, Navy, and Air Force focal points for software programs vary. However, the Joint Logistics Commanders (JLC) have established a computer resource management board and each of the Services has appointed a Computer Resources Manager (CRM). The CRMs are sources of information concerning Service-specific programs and initiatives.

Sources of expertise in software matters related to software testing, during all phases, in DoD are concentrated in the Office of the Director Defense Test and Evaluation.

Within the Services, the Development Test Commanders and Operational Test Commanders are the appropriate contacts for information concerning software testing resources.

Program management descriptions should mention any such supporting activities or clearly indicate that no additional sources of expertise are needed during the current program phase.

References for Note 11: [Klucas], [STEP2, Appendix A]

Note 12:

Over the past several years, the technology to support management of software projects has improved rapidly. Automated tools are available to estimate and track project costs, schedule tasks and monitor their progress, and implement a number of other management functions. In addition, checklists and manuals such as this one have been developed for other aspects of software acquisition. Finally, automated decision support systems with accompanying databases, local area networks, wide-band communication capabilities, workstations, and tools for supporting acquisition decision-making are available commercially.

Software managers in project offices and test organizations should be aware of the available technology and should have made a cost-benefit assessment of the desirability of using such technology (cf. Note 11).

References for Note 12: [AFOTECI], [Watt]

TEMP Section/Subsection		Checklist Questions
Part II - Program Summary		
2. Integrated Schedule: Display the integrated time sequencing of T&E for the entire program and related key events in the acquisition decision-making process. Include such events as program decision milestones, key subsystem demonstrations, test article availability, first flights, critical support resource availability, critical full-up system demonstrations, key OT&E events, first production deliveries, and initial operational capability date.	___	Are key software subsystem demonstrations included in the integrated schedule? **(Note 13)**
	___	Does the schedule show software deliveries and tool availability dates? **(Note 14)**

Note 13:

The integrated schedule should include such events as key software subsystem demonstrations and software test article availability. The schedule should also include adequate allowance for repair and retest of software, as well as time to perform the original tests. Failure to do so indicates that proper planning for critical software components has not taken place.

References for Note 13: [STEP1]

Note 14:

Support resource availability should be displayed in the integrated schedule. Software testing tools fall into this category and deserve special mention. Since these tools are themselves software, their development and acquisition are subject to many of the same risks as any other software development (cf. Note 6a). The availability dates of these tools should be included in the TEMP and/or tracked carefully by other means since a late delivery could impact the entire system development effort.

References for Note 14: [STEP1], [STEP2, Part 3]]

TEMP Section/Subsection	Checklist Questions
Part III - DT&E Outline	
This outline should discuss all DT&E in sufficient detail so that test objectives are related to the system operational concept and are clearly identified for each phase. Relate the planned testing to the critical technical issues appropriate to each phase. The following information should be included:	___ Have operational characteristics of the software that can be demonstrated during DT&E been identified? **(Note 7)** ___ Is there a plan for demonstrating appropriate operational characteristics of the software during each phase? **(Note 7)**
1. DT&E to Date. A summary of DT&E already conducted based on the best available information. Briefly describe test articles with emphasis on how they differ from planned production articles. Emphasize DT&E events and results related to required performance characteristics, critical issues, and requirements levied by earlier OSD decisions. Highlight technical characteristics or specification requirements that were demonstrated (or failed to be demonstrated). Describe how simulation models were validated.	___ Have planned levels of testing been achieved? **(Note 15)** ___ Is the documentation that reports software test results cited? **(Note 15)**

Note 15:

It should be apparent from the description of the software tests conducted and their results whether or not previous goals have been met and test objectives have been satisfied. Vague references to "successful software tests" or "no problems with the software" should not be acceptable. In order to evaluate the progress of software testing to date, there must be explicit reference to:

— a systematic, scientifically sound approach to carrying out the test

— the relationship between the systematic test approach and the test objectives for the current phase

— the results of the test

— the plans for resolution of errors.

For example, during very early program phases, unit and module testing will be conducted by contractors under government planning. The results of these tests should be maintained by the contractor. The applicable Military Standards may specify the content of these test results. If no format is contractually specified, it may be desirable to inquire into methods whereby test results can be summarized, archived, audited, and communicated to test organizations conducting higher level tests (cf. Note 10).

Systematic tests at this stage can provide indications of progress toward solving such issues as operational suitability (e.g., suitability of user interfaces and coverage of software system requirements). 100% statement coverage, complete functional tests, or random tests with specified input distributions, durations, and confidence limits are examples of systematic test approaches that can be carried out to support these objectives.

Higher level tests may cite other test approaches or the composition of lower level tests, mention results of simulations, specify goals for continuous operation under varying load, and define approaches to loading or stressing software that demonstrate the performance limits of critical software functions. In all of these cases, however, tests should not be considered performed and software issues should not be considered resolved unless the TEMP reviewer is convinced that the test methodologies cited have been carried out to completion and that the results of the tests are available for examination. Applicable Military Standards refer to Data Item Descriptions (DID's) which specify the format and contents of higher level test results.

References for Note 15: [STEP2, Part 1], [STEP3, Part 3], [STEP5, Bowen]

TEMP Section/Subsection		Checklist Questions
Part III - DT&E Outline		
1. DT&E to Date. A summary of DT&E already conducted based on the best available information. Briefly describe test articles with emphasis on how they differ from planned production articles. Emphasize DT&E events and results related to required performance characteristics, critical issues, and requirements levied by earlier OSD decisions. Highlight technical characteristics or specification requirements that were demonstrated (or failed to be demonstrated). Describe how simulation models were validated.	___	Have software deficiencies revealed by DT&E been interpreted in terms of required system characteristics and critical issues? **(Note 16)**
	___	Is the evidence clear that hardware and software failures have been properly partitioned? **(Note 16)**
	___	Have demonstrated software characteristics been highlighted? **(Note 16)**

Note 16:

Software technology is notorious for its jargon. Jargon is especially difficult to interpret in test reports. Phrases like "abend at location 11324" and "buffer overflow causing module JXAS115 to hang" describe test events very precisely and may be helpful to software engineers engaged in error location and removal -- however, these phrases are not very meaningful to system engineers. Rather than camouflage software deficiencies with overly technical descriptions, software DT&E TEMP descriptions should concentrate on technical goals, thresholds, and objectives. At each review phase, the essential questions should continue to be: Were the DT&E objectives met? If they were, with what degree of confidence were they met? If they were not met, what was the specific behavior that led to the observed anomaly?

During operational tests, the relationships between test events, software deficiencies, and unresolved test issues are more difficult to discover. In fact, the recording of test results in the operational setting may not be adequate for reconstructing the cause of a particular software failure. Therefore, a failing during OT&E is not the overly technical descriptions of software failures, but is rather the tendency to note a software anomaly with so little supporting information that traceability to critical operational T&E issues is not feasible (cf. Note 20).

Relating software T&E results to system-oriented T&E issues helps to ensure that responsibilities for deficiencies are properly allocated between hardware and software. The TEMP should provide evidence that this partitioning of errors has been the result of competent analysis. Claims that errors have been traced to either hardware or software should be dismissed unless supporting arguments can be supplied. For example, a processor chip may fail in an avionics system. However, if the software is supposed to be fault-tolerant, the error should probably be charged to the software as well as the hardware. Furthermore, the critical T&E issues should address such instances of dual responsibility.

References for Note 16: [Brown], [STEP5, Blackledge]

TEMP Section/Subsection		Checklist Questions
Part III - DT&E Outline		
1. DT&E to Date. A summary of DT&E already conducted based on the best available information. Briefly describe test articles with emphasis on how they differ from planned production articles. Emphasize DT&E events and results related to required performance characteristics, critical issues, and requirements levied by earlier OSD decisions. Highlight technical characteristics or specification requirements that were demonstrated (or failed to be demonstrated). Describe how simulation models were validated.	___	Have the differences between the software tested and the planned operational software been emphasized? **(Note 17)**
	___	Was the test environment (e.g., development, operational, or maintenance) appropriate for the characteristics to be demonstrated? **(Note 17)**

Note 17:

It is not unusual to find significant differences between the software during early stages of development and the software that will eventually be deployed. In extreme cases, the programming language may even change. More common are the many provisions that are useful for conducting such tests as unit and module tests, software integration tests, and software system function tests. These include the following:

— Test Drivers or harnesses to simulate programs that control and feed data to the software being tested

— Emulators to simulate the instructions of the operational hardware or target computer in the development environment or host computer

— Simulators for stimulating software inputs with realistic signals.

Insofar as the tests that use these techniques may be required to provide adequate verification of designs, the test results are valuable. However, care should be taken to track the course of the tested software between the current test phase and its integration into system hardware. Any changes (e.g., replacement of harnesses by operational software) that alter basic functional characteristics will require retesting at a later date. In many instances, comparison of these later tests with the current tests will be useful for locating actual differences, so arrangements for archiving the test activity (or reconstructing it) should have been made.

A parallel consideration is the nature of the test environment. While questioning the fidelity of simulations and the performance implications of target hardware emulations may be useful in other portions of the TEMP, the same issues apply to software. In addition, the software environment may have significant impact on the interpretation of the test results. For example, the assessment of whether maintainability goals have been met is complicated if the test environment does not correspond to the environment in which the software will be maintained (cf. Note 8). Similarly, other software characteristics may be influenced by even minor changes in the environment. As noted above, there should be capabilities for revisiting the software test issues addressed in the current phase of testing.

References for Note 17: [STEP2, Part 3], [STEP5, Blackledge]

TEMP Section/Subsection	Checklist Questions
Part III - DT&E Outline	
2. Future DT&E. Discuss all remaining DT&E planned, beginning with the date of the current TEMP revision. Address separately each remaining phase of DT&E, including the following for each:	
a. Equipment Description: Summary of functional capability and how it is expected to differ from the production model.	___ Have differences between the software to be tested and the planned operational software been summarized? **(Note 17)**
b. DT&E Objectives: Summary of the specific DT&E objectives to be addressed during each phase. The objectives identified should be the discrete major goals of the DT&E effort, which, when achieved, will provide solutions to critical technical issues and demonstrate that the engineering effort is progressing satisfactorily. If the OSD decision memorandum requires demonstration of specific technical characteristics in a given phase, identify those characteristics.	___ Are software DT&E test objectives traceable to required software characteristics and critical T&E issues? **(Note 7, Note 8)**
c. DT&E Events/Scope of Testing/ Basic Scenarios: Key DT&E events planned to address the objectives. In addition, describe in sufficient detail the scope of testing and basic test scenarios so that the relationship between the testing and the objectives, and the amount and thoroughness of testing are clearly apparent. Discuss RAM testing and define terms.	___ Will the planned software testing demonstrate the required characteristics? **(Note 18)**
3. Critical DT&E Items. All items the availability of which are critical to the conduct of adequate DT&E prior to the next decision point.	___ Are any new software subsystems needed for DT&E prior to the next decision point? **(Note 13)** ___ Are any new software support systems or tools needed for DT&E prior to the next decision point? **(Note 14)**

Note 18:

There should be a clear relationship between the test objectives and the software tests that are planned (cf. Note 15). Testing by "bulk" is seldom effective for software. On the other hand, statistical methods that attempt to predict the amount of software testing required are error-prone and their use should be carefully examined.

Quantitative, time-dependent goals (e.g., a fixed mean time between software failures) are difficult to demonstrate during DT&E. If quantitative information is unavailable during early phases, a second choice is to associate qualitative characteristics with quantitative goals. For example, knowing that a software reliability requirement is extremely high is usually more important than knowing that the goal is R(t)=0.997. In this case, tests should ensure that every software instruction has been executed, that all likely logic paths have been tested, that a strategy has been adopted for determining that fatal coding defects do not remain, and that all required software functions have been demonstrated. Since identifying and isolating logic paths is expensive, the adoption of such a test should be reserved for software components in which reliability is a critical issue. On the other hand, requirements may imply that the correct functioning of the software is so critical that very sensitive tests are needed. In these cases, the nature of the test and its relationship to the objective should be clearly justified.

Expense is by itself not a useful parameter in judging the effectiveness of a test for demonstrating a particular characteristic. For instance, many projects include a requirement for software endurance tests such as 25 hours of continuous operation. Such test are expensive, but they are seldom effective in uncovering software defects. Random sampling of software inputs is relatively inexpensive. However, when sampling distributions are known with a high level of confidence, results of random tests are good estimators of operational reliability.

In contrast to DT&E, quantitative time-dependent measurements are a principal result of OT&E. Even so, special care must be exercised to ensure that measurements properly reflect the software's operational suitability and effectiveness. Care must also be exercised when determining whether or not the software's contributions to overall system RAM and performance measurements have been adequately represented. For example, this determination is dependent upon the existence of enough visibility into the software during OT&E to ensure that the software contribution to OT events can be accurately assessed.

In most cases, the exact nature of the software test will not be apparent in the TEMP. The TEMP evaluator should be prepared to acquire whatever degree of detail is necessary to determine whether a given test objective can be met. A guiding principle, however, should be that ad hoc, unsystematic tests are usually not effective.

References for Note 18: [Adrion], [STEP2, Part 2], [STEP5, Bowen]

TEMP Section/Subsection

Checklist Questions

Part IV - OT&E Outline

This outline should discuss all OT&E from the earliest IOT&E through the FOT&E during initial production and deployment. Test objectives should relate the planned testing to the critical operational issues. Deficiencies in the production system should be identified. The following information should be included in similar format and detail as in the DT&E outline (Part III):

1. OT&E to Date. A summary of OT&E already conducted based on the best available information. Briefly describe test articles with emphasis on how they differ from planned production articles. Emphasize OT&E events and results related to required performance characteristics, critical issues, and requirements levied by earlier OSD decisions. Relate the test conditions and results to the operational effectiveness and suitability of the system being acquired.

___ Have software issues left unresolved during DT&E been addressed? **(Note 19)**

___ Has DT&E of operational characteristics been related to operational goals? **(Note 19)**

___ Have software deficiencies revealed by OT&E been interpreted in terms of required system characteristics and critical issues? **(Note 16)**

___ Is the evidence clear that hardware and software failures have been properly partitioned? **(Note 16)**

___ Have demonstrated software characteristics been highlighted? **(Note 16)**

___ Have differences between the software tested and the planned operational software been emphasized? **(Note 17)**

___ Was the test environment (e.g., development, operational, or maintenance) appropriate for the characteristics to be demonstrated? **(Note 17)**

Note 19:

Experience has shown that unresolved software DT&E issues are difficult to resolve during OT&E. Proceeding to dedicated operational tests with software that has not met its technical goals greatly increases the probability that expensive and time-consuming reworking will be required. On a cost basis alone, there may be an order of magnitude difference between DT&E and OT&E. Every attempt should be made to solve software problems before integration of the software and hardware (cf. Note 7).

It should also be recognized that a blurring of OT&E and DT&E may have taken place. For example, the suitability of the user interface may have already been validated during an early development test. The OT&E to date summary should review the significance of any such development tests on operational test issues. Included in this discussion should be an indication of whether the operational test objectives have been satisfied.

References for Note 19: [STEP5, Blackledge]

TEMP Section/Subsection		Checklist Questions
Part IV - OT&E Outline		
2. Future OT&E. Discuss all remaining OT&E planned, beginning with the date of the current TEMP revision. Address separately all remaining OT&E, including the following:		
a. Equipment Description: Summary of functional capability and how it is expected to differ from the production model.	___	Have differences between the software to be tested and the planned operational software been summarized? **(Note 17)**
b. OT&E Objectives: Summary of the specific OT&E objectives to be addressed during this OT&E. The objectives identified should be the discrete major goals of the OT&E effort, which, when achieved will provide solutions to critical operational issues.	___	Are software OT&E test objectives traceable to required software characteristics and critical T&E issues? **(Note 7, Note 8)**
c. OT&E Events/Scope of Testing/ Basic Scenarios: Key OT&E events planned to address the objectives. In addition describe in sufficient detail the scope of testing and basic test scenarios so that the relationship between the testing and the objectives, and the amount and thoroughness of testing are clearly apparent.	___	Will the planned testing demonstrate the required software characteristics? **(Note 18)**
Discuss the degree to which the test environment, including procedures and threat simulations, is representative of the expected operational environment. Discuss the RAM testing concept and the training and background of the operational test personnel.	___	Do the operational test analysts include software-trained personnel? **(Note 20)**
	___	Does the RAM testing concept address operational software T&E issues? **(Note 20)**
3. Critical OT&E Items. All items the availability of which are critical to the conduct of adequate OT&E prior to the next decision point.	___	Are any new software subsystems needed for OT&E prior to the next decision point? **(Note 13)**
	___	Are any new software support systems or tools needed for OT&E prior to the next decision point? **(Note 14)**

Note 20:

It is essential that some OT&E personnel have software expertise when the system contains critical software components. It is also essential to include software in the formal procedures for assigning causes to operational events.

References for Note 20: [STEP3, Part 2]

TEMP Section/Subsection		Checklist Questions
Part V - Production Acceptance Test and Evaluation (PAT&E)	___	**(Note 21)**

Note 21:

PAT&E usually presents few software issues. Notable exceptions to this rule are those instances in which software plays a critical role in manufacturing or production processes. In these cases, the relevant software may be treated as though it were a critical system component.

TEMP Section/Subsection	Checklist Questions
Part VI - Special Resource Summary	
This section provides a brief summary of the key resources for DT&E, OT&E, and PAT&E that are unique to the program.	
1. Test Articles. Identity the actual number of articles, including key support equipment, of the system required for testing in each phase and for each major type of T&E. If key subsystems are to be tested individually identify each subsystem and the quantity required. Specifically identify prototypes, pilot production, and production models.	___ Are critical software components and key subsystems identified? **(Note 13)**
2. Special Support Requirements. The special support resources required for T&E with a brief description of the steps being taken to acquire them.	___ Is there an explanation of how test tools support software test objectives? **(Note 22)**
	___ Are adequate steps being taken to acquire each tool? **(Note 23)**
	___ Do any of the software testing tools increase risk? **(Note 24)**

Note 22:

Most of the effective software testing techniques (cf. Note 18) require automated support in the form of testing tools. Some tools such as test drivers and file comparators are generic and can be used to support a variety of techniques. On the other hand, many tools are specifically designed to support a particular test methodology. The tools that have been chosen should be appropriate to carry out the planned tests. The lack of identified test tools is an indication that planned testing may be manual and therefore more labor intensive and error-prone.

References for Note 22: [STEP2, Part 3]

Note 23:

Software testing tools are in short supply. Many of the most successful tools are proprietary and must be acquired from private vendors. Several other tools have been developed in DoD labs and not widely publicized. As a last resort, the contractor, project office or test organization may develop a new tool to support a specific software test.

Reference for Note 23: [STEP2, Part 3]

Note 24:

As described in Note 6, it is possible that the technology used in a testing tool actually increases software risk. This is especially true when a tool must be developed to support a test. In that case, all of the problems of software development can occur in the acquisition of the new tool.

Other sources of risk are the technical risks associated with undemonstrated technologies and the schedule/budget risks that result from the tools that support some very sensitive test techniques. Another source of risk is the adaptation of a tool from one environment or project to another.

References for Note 24: [Bunyard]

Section 3
References and Glossary

References

[Adrion] W. R. Adrion, M. A. Branstad, and J. C. Cherniavsky, "Validation, Verification, and Testing of Computer Software," NBS Special Publication 500-75, National Bureau of Standards.

[AFOTECI] "Software OT&E Guidelines," Volume I. Software Test Manager's Handbook, Air Force Operational Test and Evaluation Center.

[AFOTECIII] "Software OT&E Guidelines," Volume III. Software Maintainability Evaluator's Handbook, Air Force Operational Test and Evaluation Center.

[Brown] David R. Brown, "Software Needs Top-Down Management," Concepts: The Journal of Defense Systems Acquisition, Volume 5, Number 4 (Autumn, 1982), pp. 202-218

[Bunyard] MG Jerry Max Bunyard and James Mike Coward, "Today's Risks in Software Development -- Can They Be Significantly Reduced?" Concepts: The Journal of Defense Systems Acquisition, Volume 5, Number 4 (Autumn, 1982), pp. 73-94.

[Chou] W. Chou (editor), Computer Communications, Volume 1 -- Principles, Prentice-Hall, 1983.

[Grove] H. Mark Grove, "DoD Policy for Acquisition of Embedded Computer Resources," Concepts: The Journal of Defense Systems Acquisition Management, Volume 5, Number 4 (Autumn, 1982), pp. 9-36.

[IBM] "Application Development and Maintenance Measurement and Analysis," IBM Corporate Information Systems and Administration Guideline, No. 103 (February 26, 1982).

[Klucas] Caspar H. Klucas, Larry A. Fry, John W. Barnes, Matthew Fisher, "Joint Service Policy and Standards", Concepts: The Journal of Defense Systems Acquisition Management, Volume 5, Number 4 (Autumn, 1982), pp. 191-201

[Meyers] G. J. Meyers, Software Reliability: Principles and Practices, Wiley Publishers, 1976.

[Perlis] Alan Perlis, Fred Sayward and Mary Shaw, Software Metrics: An Analysis and Evaluation, MIT Press, 1981.

[Redwine] Samuel T. Redwine, et al., "DoD Related Software Technology Requirements, Practices, and Prospects for the Future," IDA Paper P-1788, June 1984, Institute for Defense Analyses.

[STEP1] R. A. DeMillo and R. J. Martin, "Software Test and Evaluation Project Phases I and II Final Report: Volume 1. Report and Recommendations," Director Defense Test and Evaluation.

[STEP2] "Software Test and Evaluation Project Phases I and II Final Report: Volume 2. Software Test and Evaluation: State-of-the-Art Overview," Director Defense Test and Evaluation.

[STEP3] "Software Test and Evaluation Project Phases I and II Final Report: Volume 3. Software Test and Evaluation: Current Defense Practices Overview," Director Defense Test and Evaluation.

[STEP5] "Software Test and Evaluation Project Phases I and II Final Report: Volume 5. Report of Expert Panel on Software Test and Evaluation" Director Defense Test and Evaluation. (Michael A. Blackledge, "Acquisition Problems Influencing Software Development and Operational Testing," pp. 116-122; John B. Bowen and Marion F. Moon, "Experience in Testing Large Embedded Software Systems", pp. 57-64; William P. Nelson, "Software Quality Assurance and Acquisition Policy", pp. 105-111; Marilyn J. Stewart, "Software Testing Standards - Policy and Applications", pp. 99-104)

[Watt] Charles K. Watt, "The Evolution of Information Systems," Journal of Test and Evaluation, Volume V, Number 2 (April/July, 1984), pp. 9-13

Glossary

Automatic Data Processing (ADP)

In its most general usage, this refers to any use of computers to process information. Usually, however, automatic data processing is contrasted with MCCR applications; for example, personnel and payroll applications are ADP while weapon systems are MCCR applications.

Compile

The process of translating a computer program from one programming language (the source language) to another (the object language).

Complete Functional Test

The process of demonstrating that each functional requirement is satisfied by the software.

Critical Software Component

Any portion of a computer program, any computer program or any collection of computer programs that fulfills a requirement for a Key Function as described in the TEMP.

Critical Software T&E Issue

Any aspect of the software system's capability that must be questioned before a system's overall worth can be estimated.

Data Base Manager

Software that is used to control and access large amounts of data that are organized into data bases.

Data Item Description (DID)

A document that contains the format and content preparation instructions for a contract deliverable consisting of data generated under work tasks described within military standards.

Emulator

A program or device that simulates the execution of one computer by another one.

Environment

As applied to software, an environment is the collection of methodologies, tools, machines and management practices by which software is engineered. Sometimes the environment is embodied in an extensive piece of software; more frequently, the environment is a combination of manual and automated procedures and methodologies. Environments may be distinguished by lifecycle phase (e.g., development environment or PDSS environment).

Evolutionary Acquisition

The structuring of the acquisition process to accommodate evolutionary or incremental development.

Goal

The desired or expected value of a required software characteristic.

Government Furnished Equipment (GFE)

As applied to software, GFE refers to any software tool or system that is supplied by the Government to a contractor.

High Order Language (HOL)

Any programming language that removes machine dependencies or permits the expression of programming constructs in more natural terms than would otherwise be possible. Common high order languages include Fortran, Jovial, Ada, C, Pascal, and CMS-2.

Host Computer

The computer on which the software is being developed or tested.

Incremental Development

The process of designing, implementing, testing and delivering a software product in increasingly complete increments.

Lifecycle

The structuring of the phases and activities of the design, implementation, and operation process as a function of time. No single lifecycle adequately describes all software products. Model lifecycles are contained in appropriate Service standards and the design documents and standards of many software developers.

Maintenance

As applied to software, maintenance refers to the process of correcting errors, modifying designs, and adding new capabilities. To avoid confusion between corrective maintenance (correcting errors) and the re-engineering of software products, the entire activity is sometimes referred to as post-development or post-deployment software support (PDSS).

Mature Software

Software which has demonstrated through extensive usage, testing and repair of defects that it meets design and user requirements. An indication of software maturity is the extent to which it is modified after new tests.

Mean Time Between Operational Mission Failure (MTBOMF)

An operational mission failure is any system condition observed during system operation under operational conditions that results in a failure to meet one or more system mission objectives. These failures may be due to hardware, software, or operator failures.

Mission Critical Computer Resources (MCCR)

See Note 1.

Mission Critical Software

Software that implements MCCR functions that are essential to the performance of the system or mission objectives.

Module Testing

Often used interchangeably with unit testing. More often, module tests refer to tests of independently compiled software routines against technical specifications.

Post Deployment Software Support or Post Development Software Support (PDSS)

See Maintenance.

Preplanned Product Improvement (P3I)

An acquisition and design strategy that involves the scheduled and planned enhancement of a system or product.

Proprietary Software

Software owned by an individual or organization that has placed restrictions on the use or disposition of the software by others who do not own it. These restrictions are usually imposed by the commercial sector through licenses or other agreements that detail the rights to which the licensee is entitled.

Prototyping

The process of producing an experimental version of a software system or portion of a software system in order to evaluate one or more aspects of the design.

Random Test

The process of supplying software input values chosen at random by sampling from a fixed distribution.

Required Software Characteristics

Parameters that are the primary indicators of the software's conformance to written specifications (the technical characteristics) and operational suitability and effectiveness (operational characteristics). Ideally, these characteristics should be quantitatively specified by the range of minimally acceptable (threshold) and desired (goal) values of the parameter.

Reusable Software

Existing software that can be inserted into use on a given system with little or no modification.

Software Engineering

The practice of designing and constructing software products using disciplined, controlled, and monitored engineering techniques.

Software Intensive

See Note 2.

Software Quality

The extent to which the software meets technical specifications, and user needs and expectations. The totality of features and characteristics of the software that bear on its ability to satisfy given needs.

Software Requirements

The statements of software systems capability that are the basis of software design. The mechanisms for describing software requirements vary among the Services. For details see MIL-STD-490 (Specification Practices), DoD-STD-1679 (Weapon System Software Development), and DoD-STD-2167 (Defense System Software Development).

Software Tool

A computer program that provides automated support for the development of other software products. Typical tools include compilers, editors, debuggers, testing tools, librarians, mail facilities, and various design aids.

Statement Coverage

The number or percentage of program statements that have been executed during testing. The utility of 100% statement coverage is that there may be latent defects in coding lines that have not been executed during a test.

Stress Testing

The execution of tests that attain or exceed maxima defined by one or more software requirements.

Systematic Software Test

Any software test that involves the usage of a scientifically sound test approach. Systematic approaches should be contrasted with ad hoc tests that may specify procedures and activities only.

Target Computer

The operational computer.

Test Drivers

Software that is used to initiate or control the execution of the software components being tested. The most common examples of test drivers occur during unit and module tests when software subsystems are controlled by generic test drivers that simulate subsystem calling sequences and provide for data transfer in and out of the subsystem.

Test Harness

Test Driver.

Threshold

The minimally acceptable value of a required software characteristic.

Unit Testing

Testing of small, logically coherent pieces of software (such as subroutines) against technical specifications.

User Interface

The (usually electronic) interface between the human and the software.

INDEX

acquisition policy
7, 52
Ada
17, 29, 54
Ada Joint Program Office
29
Air Force Language Control Facility
29
artificial intelligence
13
automatic data processing
7, 9, 53
availability
7, 11, 20, 22, 23, 24, 27, 32, 33, 40, 44
Brooks Act
7
CAD/CAM
9
command and control
7
communication
15, 31
compiler validation
17
computer graphics
9
Computer Resources Manager
31
critical DT&E items
40
critical interfaces
9
critical OT&E items
44
critical software component
1, 11, 21, 27, 33, 45, 48, 53
cryptologic
29
data base
53
data item description
35, 53
data-sharing
7
Director Defense Test and Evaluation
31, 52
Director, Computer Software and Systems
31
distributed data processing
13

DoD Directive 5000.3
1
DoD Directive 5000.29
7
DoD Security Center
7
DoD-STD
56
DT&E objectives
37, 40
DT&E outline
42
DT&E to date
34, 36, 38
emulator
39, 53
endurance tests
41
error masking
21
error partitioning
21
fault-tolerance
7
Federal Software Testing Center
29
file comparators
49
functional tests
35
future DT&E
40
future OT&E
44
GFE
17, 54
goals and thresholds
20, 21, 22, 24, 27
harnesses
39
high order language
17, 54
human interfaces
9
industry standards
15
integrated schedule
32, 33
intelligence
7, 13
inter-agency agreements
7
interoperability
15

Joint Logistics Commanders
31
Jovial
17, 29, 54
Key Function
11, 12, 15, 53
logic paths
41
maintainability
20, 22, 23, 24, 27, 39, 51
maintenance
9, 23, 27, 51, 55, 56
mature
13, 17, 55
MIL-STD
56
military standards
35, 53
mission critical
1, 6, 7, 9, 11, 15, 21, 55
mission critical computer resources
9, 11, 55
mission objectives
7, 9, 55
mission/function matrix
12, 13
MTBOMF
55
MTTR
27
National Bureau of Standards
51
networking
9
non-standard acquisition
19
operational environment
4, 23, 27, 44
operational suitability
23, 35, 41, 56
OT&E objectives
44
OT&E outline
42, 44
OT&E personnel
45
OT&E to date
42, 43
PAT&E
47, 48
performance
13, 17, 19, 20, 22, 23, 24, 25, 34, 35, 36, 38, 39, 41, 42, 55

Post Deployment Software Support
23, 55
Product Engineering Services Office
29
programming environments
29
proprietary
17, 49, 56
prototyping
13, 19, 56
quality
13, 52, 57
RAM
21, 40, 41, 44
random tests
35, 41
real-time
7
reliability
19, 20, 22, 23, 24, 27, 41
required operational characteristics
20, 21, 22, 24
required software characteristics
21, 23, 27, 56
required technical characteristics
20, 22, 24
risk
1, 7, 9, 11, 13, 15, 17, 26, 27, 49
robustness
23
simulation
9, 13, 34, 36, 38
simulators
23
software deficiencies
36, 37, 42
software design methodologies
13
software engineering
9, 16, 18, 56
software intensive
1, 7, 8, 9, 11, 56
software manager
29, 31
software reuse
13
software requirements
9, 11, 56, 57
software subsystem demonstrations
33
software T&E issues
7
software test article
33

software testing tools
33, 48, 49
special resource summary
48
statement coverage
35, 57
stress testing
57
support software
9, 17, 29, 48
T&E issue
7, 26, 27, 37, 53
TEMP
1, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 20, 21, 22, 24, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 46, 48, 53
test articles
34, 36, 38, 42, 48
test drivers
49, 57
test objectives
35, 40, 41, 42, 43, 44
time dependent reliability
27
training
44
unique characteristics
6, 8, 10, 16, 18
unresolved software DT&E issues
43
weapon
7, 53